PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

D4

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : H04K 1/02 | A2

(11) International Publication Number: WO 99/45670

(43) International Publication Date: 10 September 1999 (10.09.99)

(21) International Application Number: PCT/IB99/00366

(22) International Filing Date: 3 March 1999 (03.03.99)

(30) Priority Data:
09/035,430 5 March 1998 (05.03.98) US

(71) Applicant: KONINKLIJKE PHILIPS ELECTRONICS N.V. [NL/NL]; Groenewoudseweg 1, NL–5621 BA Eindhoven (NL).

(71) Applicant *(for SE only)*: PHILIPS AB [SE/SE]; Kottbygatan 7, Kista, S–164 85 Stockholm (SE).

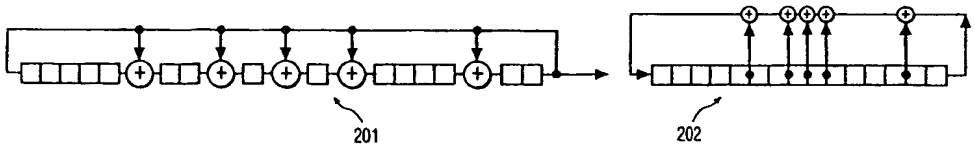
(72) Inventor: MEDLOCK, Joel; Prof., Holstlaan 6, NL–5656 AA Eindhoven (NL).

(74) Agent: DEGUELLE, Wilhelmus, H., G.; Prof. Holstlaan 6, NL–5656 AA Eindhoven (NL).

(81) Designated States: CN, JP, KR, European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
Without international search report and to be republished upon receipt of that report.

(54) Title: MASK GENERATING POLYNOMIALS FOR PSEUDO–RANDOM NOISE GENERATORS



(57) Abstract

A CDMA baseband modem having a PN generator with significant reduction in stored masks is disclosed. Various performance matrices set forth alternative combinations of sequence generation through three parameters: storage, software, and time. Embodiments of this implementation are presented with corresponding hardware complexities. Masks are used to move to a new phase offset sequence. Instead of storing 2^N masks, less than N masks are required to be stored in ROM because of mask calculation intelligence carried out in the software/software control. Masks are calculated with a mask generating polynomial based upon any degree of characteristic polynomial of a PN generator. Masks for the mask generator are only stored which result in phase shifts of powers of two. The characteristic polynomial of the pseudo–random noise generator and the mask generator are Galois and Fibonacci polynomials.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

Mask generating polynomials for pseudo-random noise generators.

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates to Code Division Multiple Access (CDMA) personal communications systems, and particularly to pseudo-random noise (PN) generators for a mask generating characteristic polynomials in a CDMA communications system.

2. Discussion of Background Art

Code Division Multiple Access (CDMA) is commonly used for bandwidth spreading of a digital signal in wireless communications technology such as PCS. Instead of frequencies or time slots, CDMA uses mathematical codes to transmit and distinguish between multiple wireless communication channels. Standard radio receivers separate stations and channels by filtering in the frequency domain. CDMA receivers, on the other hand, separate communication channels by pseudo-random modulation that is applied and removed in the digital domain. Frequency reuse, therefore, distinguishes CDMA's high spectral efficiency. Because the conversations are distinguished by digital codes, many users share the same bandwidth simultaneously. Bandwidth is much wider than that required for simple point-to-point communications at the same data rate because noise-like carrier waves spread the information contained in a signal of interest.

In CDMA, many stations, each having its own unique code, simultaneously transmit their signals which are discriminated from all of the others. A cross-correlator accomplishes this discrimination by using as a reference the code corresponding to that of the desired transmission. CDMA codes are particularly selected to have good cross-correlation properties so that interference from simultaneous transmissions of other signals is reduced.

Before CDMA, spread spectrum communications had been used for many years to encrypt military communications. It is difficult to interfere with or intercept a CDMA message because of its spread signal, and thus its chief strength was to resist enemy jamming and to provide secure communication. Applications to civilian mobile communications were proposed in the 1940's, but practical applications in the civilian communications market did not take place until forty years later. A first field test was demonstrated in 1991, and the

CONFIRMATION COPY

technology was tested, standardized, and deployed in less than seven years. Interim Standard-95 (IS-95) became a digital cellular standard after adoption in 1992, and approval in 1993, by TIA (Telecommunications Industry Association). IS-95 systems divide the radio spectrum into carriers which are 1.250 MHz wide.

Spread spectrum communication provides increased bandwidth in a limited frequency system, and has additional advantages including extended range and more communication security. In CDMA, a narrowband message signal is multiplied by a spreading signal, which is a pseudo-random noise code sequence having a rate much greater than the data rate of the message. The pseudo-random code sequences distinguish individual conversations.

CDMA artificially increases the bit data rate by breaking each original signal bit into a number of sub-bits called "chips." For an increase factor of 10, each bit of the original signal is divided up into 10 separate bits, or "chips," to thereby increase the data rate by 10. The bandwidth is also increased by a factor of 10.

The pseudo-random noise code is a sequence of high data rate bits ("chips") ranging from -1 to 1 (polar) or 0 to 1 (non-polar). In "direct sequence" spread spectrum devices, the pseudo-random noise code of a -1 (or 0) data bit is the inverse of the pseudo-random noise code of a 1 data bit, while the pseudo-random noise code for these data bits of "indirect sequence" spread sequence devices are other than inverse codes. A direct sequence code of length N = 4 might spread a polar +1 bit to {+1, -1, +1, -1} and a polar -1 bit to an inverse{-1, +1, -1, +1}, while an indirect sequence code of length N = 4 might spread a polar +1 bit to {+1, -1, +1, -1} and a polar -1 bit to {+1, +1, +1, -1}.

"Chips" thus refers to the number of small data bits in the PN code that are added to each single bit in the original signal. This is performed by multiplying the original modulated signal by this high data rate PN-code results (for polar), or with an exclusive OR operation in binary arithmetic (for non-polar). A wider bandwidth signal is produced proportional to the number of "chips." The receiver then removes the PN-code to obtain the original signal by multiplying with a replica code sequence (for polar).

The pseudo-random code is a complex pattern that ensures the receiver does not accidentally synchronize with another signal. A binary pseudo-random noise code of length N creates 2^N possible codes. To minimize interference between callers, however, these codes must be orthogonal to one another. (Signals are completely orthogonal if they differ in exactly half of their bit sequences.) There are only N orthogonal spreading sequences of length N.

A CDMA cellular phone call begins with a standard rate of 9600 bits per second. This is then spread to a transmitted rate of about 1.23 Mbits per second. Spreading applies the digital pseudo-random noise code associated with a particular cell user to the data bits. The data bits are transmitted along with the signals of all of the users in that cell. Codes are removed from the desired signal when the signal is received, thereby separating the users and returning the call to a rate of 9600 pits per second.

Spread signal multiple access systems transmit the entire signal over a bandwidth that is much greater than required for standard narrow band transmission in order to increase signal-to-noise (S/N) performance. Increasing the transmitted signal bandwidth results in an increased probability that the received information is correct for channels having narrowband noise. Because each signal is a compilation of many smaller signals at a fundamental frequency and its harmonics, increasing the frequency results in a more accurate reconstruction of the original signal.

The performance increase for very wideband systems is referred to as "process gain," G. This term describes the fidelity of the received signal gained at the cost of bandwidth, and is defined as W/R, where W is the spread bandwidth and R is the data rate. Errors introduced by a noisy channel are reduced to any desired level without sacrificing information rate transfer using Shannon's equation of channel capacity:

$$C = W \log_2 (1 + S/N)$$

where C is the channel capacity in bits per second, W is the bandwidth, and S/N is the energy per bit divided by noise power. It is clear form this equation that increasing the bandwidth permits a decrease in the S/N ratio without decreasing the bit error rate. In IS-95A CDMA,

$$W/R = 10 \log (1.2288 \text{ MHz} / 9600\text{Hz}) = 21 \text{ dB},$$

for a 9600 bps data rate.

U.S. Patent No. 5,228,054 to *Rueth et al* discloses a prior art pseudo-random noise generator that augments the length of a PN sequence by 2^N. The *Rueth et al* patent represents a hardware-only approach by providing a sequence augmenting circuit with a linear sequence shift register to augment a PN sequence length by inserting additional "chips" into the sequence at a predetermined position. The LSSR generates a PN sequence of length $2^N - 1$, and the augmenting circuit inserts at least one additional "chip" to augment the sequence length to 2^N.

SUMMARY OF THE INVENTION

The present invention involves several implementations of a PN generator for a short code sequence with consideration to masking in a CDMA communications system, particularly for a modem. Various performance matrices set forth alternative combinations of sequence generation through three parameters: storage, software, and time. By varying these three parameters, it becomes possible to optimize system performance while reducing cost. Instead of storing 2^N masks to move to a new phase offset, only N masks are stored in ROM due to the ability of software to calculate masks with a mask generator. Masks are only stored in ROM which result in a phase shift of a power of two.

The invention consists of a PN Generator and a Mask Generator. A PN Generator creates the pseudo-random noise code, and a Mask Generator creates masks to shift the phase of the PN sequence. The method for determining a characteristic polynomial for a Mask Generator depends upon a characteristic polynomial of a PN Generator. These two polynomial characteristics are complementary. PN Generators are of "Galois configuration" or "Fibonacci configuration," and the corresponding Mask Generator, therefore, is of "Fibonacci configuration" or "Galois configuration," respectively. For a characteristic polynomial for the PN Generator of the form:

$$f(x) = 1 - \sum_{i=1}^{N} c_i x^i$$

a corresponding Mask Generator characteristic polynomial is:

$$h(x) = 1 - \sum_{i=1}^{N} c_{N-i} x^{N-i}$$

The initial state of a Mask Generator is determined by an output bit from the PN Generator.

The Mask Generator of the present invention permits the modem to calculate a required mask for any phase offset (e.g., from zero to $2^{15} - 1 = 32767$). Faster switching of phases between PN sequences is possible due to the reduction in number of masks required to be stored in ROM. Depending upon particular implementation, masks are at least partially computed with a software algorithm.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 shows a first embodiment prior art masking circuit;

Figures 2(a) and 2(b) illustrate two embodiments of pseudo-random noise and masking generators;

Figure 3 illustrates a second embodiment of a pseudo-random noise generator;

Figure 4 shows a third embodiment of a pseudo-random noise generator;

Figure 5 illustrates a fourth embodiment of a pseudo-random noise generator; and

Figure 6 shows a fifth embodiment of a pseudo-random noise generator.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

The present invention determines a characteristic polynomial for a mask generator as a complementary form for a characteristic polynomial of a PN generator. The characteristic polynomials have complementary forms of a "Galois configuration" or a "Fibonacci configuration." For a characteristic polynomial for the PN sequence generator of the form:

$$f(x) = 1 - \sum_{i=1}^{N} c_i x^i$$

a corresponding Mask Generator characteristic polynomial is:

$$h(x) = 1 - \sum_{i=1}^{N} c_{N-i} x^{N-i}$$

A linearly recursive formula for an I sequence generator is i(n) = i(n - 15) + i(n - 10) + i(n - 8) + i(n - 7) + i(n - 6) + i(n - 2). Therefore, an I pseudo-random noise polynomial sequence has a characteristic polynomial of $f(x) = x^{15} + x^{13} + x^9 + x^8 + x^7 + x^5 + 1$, and the corresponding mask generator has a polynomial of the form $h(x) = x^{15} + x^{10} + x^8 + x^7 + x^6 + x^2 + 1$.

Each communication channel in CDMA has a particular code. Thus, many parties have the ability to communicate simultaneously since each communication channel uses a different code. In order to suppress the other codes, integration is required to test for the presence of the desired code. Integration over a pseudo-random noise code generally produces a result of zero because a truly random code is equally likely to be a positive or negative binary digit. The correlation of these random bits averages to zero. This is not the situation, though, if the transmitter and receiver codes are identical. When the transmitter and

receiver use the same code, the integration results in a non-zero value. All communication channels are thereby suppressed except for the desired communication channel whose code is shared by the parties.

A CDMA baseband modem requires 1.2288 MHz PN sequences of length 2^{15} called "short codes." A shifted short code sequence has a property of orthogonality. Non-shifted sequences with identical PN codes produce a non-equal number of one's and zero's (e.g. a sequence of "1111" or "0000") when exclusively-ORed with one another; sequences with different PN codes produce an equal number of one's and zero's when exclusively-ORed with one another. Multiplying a sequence shifted with respect to another sequence integrates to zero over time (referred to as "spreading"). Multiplying a sequence not shifted with respect to another sequence integrates to a ramp function (referred to as "despreading").

Ideally, a sequence is integrated over its entire length to determine whether the result is a zero or non-zero number. However, integration over the entire PN sequence is impractical in IS-95. Instead, the CDMA modem integrates over a section of the PN sequence, offsets the PN phase, and performs a new integration to determine whether a shifted sequence is present.

A pseudo-random noise sequence is generated by a linear feedback shift register (LFSR), or a linear sequence shift register (LSSR). This sequence has a characteristic sequence rate and the data has a characteristic data modulation rate. The data is exclusively ORed with the pseudo-random sequence, so that synchronization only occurs for common factors. A PN sequence rate of 1.2288 MHz with a data modulation rate of 9600 bits per second produces 128 PN "chips" per information bit, and a data rate of 4800 bits per second corresponds to 250 PN "chips" per information bit. The coincidence between the data and the PN sequence occurs every 128 or 256 repetition intervals of the PN sequence. It is desirable to make the length of PN sequence to be a power of two (i.e. 2^N) so that the repetition interval of the PN sequence and the data rate happens more frequently.

There are several representations for the operation of a Linear Feedback Shift Register. One representation is in powers of α. If each LFSR state is considered a vector, then for any primative polynomial the total number of unique vectors is equal to one minus the polynomial's degree N raised to a power of two. Each vector is constructed from the binary field GF(2) and form the elements of a larger field $GF(2^N)$, where N is the degree of the polynomial. Thus a polynomial of $f(X)=1+X^5+X^7+X^8+X^9+X^{13}+X^{15}$ has $2^{15}-1$ elements (vectors) which belong to the field $GF(2^{15})$. Vectors can be represented as an ordered sequence of m components called m-tuples. Each m-tuple in the vector space $GF(2^{15})$ can be composed

of linear combinations of the unit vectors of $\alpha^0,\alpha^1,\alpha^2,...,\alpha^{14}$. Every vector can be determined by multiplication's by α and hence addition of exponents. Using the previous polynomial's relation that $\alpha^{15}=1+\alpha^5+\alpha^7+\alpha^8+\alpha^9+\alpha^{13}$, any powers of α greater than 15 can be represented in terms of the powers of α less than 15. (Note the addition is modulo-two). For example $\alpha^{17}=\alpha^{15}\alpha^2=(1+\alpha^5+\alpha^7+\alpha^8+\alpha^9+\alpha^{13})(\alpha^2)=1+\alpha^2+\alpha^5+\alpha^8+\alpha^{10}+\alpha^{11}+\alpha^{13}$.

The operation of multiplication by α is identical to a linear transformation of a vector (m-tuple). Thus another representation for the operation of an LFSR is with linear transformations. The particular linear transform is in the form of a matrix multiplication and models both the operation of the shift register and the feedback operation. For the Galios LFSR, the operation is a left-hand side multiplication with a column vector. If a v is a 15-tuple column vector, the matrix multiplication can be written as:

$$\begin{bmatrix} c_{14} & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ c_{13} & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ c_{12} & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ c_{11} & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ c_{10} & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ c_{9} & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ c_{8} & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ c_{7} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ c_{6} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ c_{5} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ c_{4} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ c_{3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ c_{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ c_{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ c_{0} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} v_{14} \\ v_{13} \\ v_{12} \\ v_{11} \\ v_{10} \\ v_{9} \\ v_{8} \\ v_{7} \\ v_{6} \\ v_{5} \\ v_{4} \\ v_{3} \\ v_{2} \\ v_{1} \\ \end{bmatrix} = M.v = \alpha v = v'$$

where the c_n represent 0 or 1 for the coefficients of the previous generating polynomial. The resulting column vector **v'** is identical to iterating the Galois LFSR by one clock cycle. Thus a PN sequence of length 2^{15}-1 that is generated with an LFSR will contain 2^{15}-1 distinct 15-tuples. Moreover, after 2^{15} -1 matrix multiplications with an initial m-tuple, the original m-tuple will result. This operation extends to any polynomials of any degree.

The circuit of Figure 1, referred to as a "mask" circuit, performs the inner product (or dot product) of two vectors which belong to the set V_{15} or vector space over the field GF(2^{15}). If **u** and **v** are vectors in the vector space V_n, the scalar product is

$$\mathbf{u} \bullet \mathbf{v} = \sum_{i=o}^{N-1} u_i \bullet v_i = u_0 \bullet v_0 + u_1 \bullet v_1 + \ldots + u_{N-1} \bullet V_{N-1}$$

where $u_i \bullet v_i$ is carried out in modulo-two multiplication and u_i+v_i is carried out in modulo-two addition. The modulo-two multiplication is implemented with an "AND" gate. The modulo-two multiplication is implemented with an "Exclusive-Or" gate. Thus the inner product between two vectors in the vector space forms a scalar. In this case the scalar is an element over GF(2). The inner product has the mathematical properties of:

(i) $\mathbf{u} \bullet \mathbf{v} = \mathbf{v} \bullet \mathbf{u}$ commutative

(ii) $\mathbf{u} \bullet (\mathbf{v}+\mathbf{w}) = \mathbf{u} \bullet \mathbf{v} + \mathbf{u} \bullet \mathbf{w}$ distributive

(iii) $(\mathbf{au}) \bullet \mathbf{v} = (\mathbf{u} \bullet \mathbf{v})$ associative

The operation of an inner product is identical to the inner product of a row vector multiplied with a column vector resulting in a scalar. The resulting scalar is an element over GF(2) which directly maps to a bit within the output PN sequence. The mapping is one to one and each possible vector is associated with a unique offset in the sequence. The purpose of mask generating polynomials is to effectively predict the required vector for any given offset.

The required mask vector for a zero offset has a trivial case of N-1 "zeros" and a single "one" corresponding to the output tap of the LFSR. For example, if the LFSR output is from the fifteenth bit of a shift register, the mask for a zero offset is all zeros except for a one in the fifteenth position:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \\ c_8 \\ c_9 \\ c_{10} \\ c_{11} \\ c_{12} \\ c_{13} \\ c_{14} \end{bmatrix} = c_{14}$$

The above notation assumes the LFSR output is taken from the fifteenth output bit. The c_n coefficients correspond to any particular n-tuple within the vector space V_n.

The operation of a mask circuit combined with an LFSR is given in the following form where **u** is row vector and **v** is a column vector:

$$\mathbf{u}\,(\mathbf{M}^n \cdot \mathbf{v}) = x^n$$

Each iteration of the LFSR is equivalent to multiplication by **M**. The output x^n is the result of forming the inner product with the resulting vector and the "mask" vector **u**. If the mask vector **u** is static, iterations of the LFSR (or successive multiplications by **M**) will generate a bit sequence of x^n which is offset from the bit sequence produced by **v** alone.

The process of determining the correct "mask vector" for any arbitrary shift can be seen by using the associativity property. Thus the identical output x^n can determined with the following relation:

$$(\mathbf{u} \cdot \mathbf{M}^n)\,\mathbf{v} = x^n$$

The above expression assumes that **u** is the vector corresponding to the zero shift. The vector for any α^m offset is equivalent to setting N=m and performing N right hand side matrix multiplication's with a row vector. The identical sequence to the previous x^n can be

generated with successive row-matrix multiplications while keeping the vector **v** constant. This is the basic process for generating arbitrary "mask" vectors. Upon inspection of the row-matrix multiplication, the operation is identical to a "Fibonacci" LFSR. The coefficients of the Fibonacci LFSR are directly related to the generating polynomial but reversed.

$$\begin{bmatrix} a_{n-1} & a_{n-2} & \cdots & a_{n-14} & a_{n-15} \end{bmatrix} \begin{bmatrix} c_{14} & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ c_{13} & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ c_{12} & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ c_{11} & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ c_{10} & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ c_{9} & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ c_{8} & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ c_{7} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ c_{6} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ c_{5} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ c_{4} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ c_{3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ c_{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ c_{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ c_{0} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} =$$

$$(c_{14}a_{n-1} + c_{13}a_{n-2} + \ldots + c_0 a_{n-15}, a_{n-1}, a_{n-2}, \ldots, a_{n-15})$$

To efficiently compute any "mask vector", iterations of the Fibonacci LFSR must be computed efficiently. Although it is possible to generate masks by iterating the Fibonacci LFSR for N times starting from the zero shift vector, more efficient algorithm exist. In the same way the **u** vector is a mask for the **Mv** Galois LFSR, the **v** vectors are masks for the **uM** Fibonacci LFSR. Thus the Fibonacci LFSR output can be offset as well. By storing a small set of Galois vectors to shift the Fibonacci LFSR, an efficient mask generating algorithm can be determined. Moreover, if the Galois vectors are chosen to correspond to shifts by powers of two, then the bits of a binary representation of an offset will correspond directly to the small set of stored vectors. These Galois vectors can be used to shift the Fibonacci LFSR to generate any required mask vector.

Figure 1 illustrates a first prior art embodiment of masking circuit 100. A sequence of fifteen AND gates 101a-o each has an input for receiving a data bit from bus 102

and a mask bit from bus 103. The output of each of AND gates 101a-o is coupled to one of fifteen exclusive OR gates 104a-o. The first exclusive OR gate 104a has one input coupled to AND gate 101a and one input coupled to ground. The last exclusive OR gate 104o has an output providing a SHIFTED OUTPUT sequence. A 15 bit word is expected for a desired phase offset.

When these parameters are incorporated into the hardware, a table of 512 x 15 bit words produces inaccuracies up to 64 chips. Compensation is achieved by "slewing," which is defined as a process of clocking the PN generator at a rate other than 1.2288 MHz. "Slewing forward" refers to clocking at a rate faster than 1.2288 MHz, while "slewing backward" is clocking at a rate slower than 1.2288 MHz. The number of required clock cycles to change the phase offset is (1 + 64) given unidirectional slewing. The software complexity for this approach is zero since no intelligence is used in loading masks. The total ROM storage is 15,360 bits.

Table 1 shows the masking performance of this first embodiment. The columns for H/W slewing cycles and computation iterations are separate because these algorithms are done in parallel. The H/W slewing cycles do not necessarily correspond to chips since the H/W clock cycles are done at a faster chip rate if a faster clock is multiplexed.

Table 1

	ROM bits	Hardware Complexity	Hardware slewing cycles (max)	Computation Iterations
I-PN Code	7680	30*	64**	0
Q-PN Code	7680	30*	64**	0

*Estimate is a combination of AND and XOR gates. Since all masking techniques require a mask register, its complexity is not included.

**Given unidirectional slewing

In Table 1, the "H/W complexity" refers to the mask circuit, and additional logic creates extra zeros in the PRIMARY OUTPUT and SHIFTED OUTPUT data streams. The embodiment of Figure 1 maintains a particular PN offset, and the mask creates a relatively shifted sequence. One advantage of this architecture is that it permits rapid switching between a shifted and non-shifted sequence, which property has application to inter-frequency hard handoff in IS-95-B by permitting the mobile to quickly return from the hard handoff.

Rake finger performance is also affected by the number of clock cycles required to change a phase offset. A Rake finger in an IS-95 system is required to demodulate to a new base station, and the finger needs to quickly change its PN phase offset to do this. Therefore, system performance is affected when a Rake finger switching is delayed, and this delay depends upon the speed of changing the PN phase offset. Moreover, a long delay results in a loss of information from the newly assigned PN offset.

Figures 2(a) and 2(b) illustrate two embodiments of PN and mask generators. In Figure 2(a), PN Generator A 201 has a Galois configuration and corresponding Mask Generator A 202 has a Fibonacci configuration. The initial state of Mask Generator A 202 is determined by the output bit of PN Generator A 201. When the output bit of PN Generator A 201 is the least significant bit (LSB), the initial state of Mask Generator A 202 is 0x0001. The 0x0001 state vector generates a zero shifted sequence in a masking circuit 100.

Mask Generator B may be used to generate masks for PN Generator B. The initial condition of 0x4000 corresponds to a zero offset sequence. The 0x4000 mask may be used independently of the state of PN Generator B to always result in a zero offset sequence. One iteration of Mask Generator B results in a single bit offset sequence. Sample data of a Mask B Generator State versus PN sequence shift is shown in Table 2. . A similar table may be constructed for a PN Generator A and Mask Generator A.

Table 2

State of Mask Generator B	Shift relative of PN Generator B sequence
0 x 4000	0
0 x 2000	1
0 x 1000	2
0 x 0800	3
0 x 0400	4
0 x 0200	5
0 x 0100	6
0 x 0080	7
0 x 0040	8
0 x 0020	9
0 x 0010	10
0 x 0008	11
0 x 0004	12
0 x 0002	13
0 x 0001	14
0 x 51D0	15
0 x 28E8	16
0 x 1474	17

Figure 2(b) correspondingly shows a PN Generator B 251 having a Fibonacci configuration, and Mask Generator B 252 having a Galois configuration. Both the PN Generators and Mask Generators of Figures 2(a) and 2(b) are typically linear feedback shift registers or linear sequence shift registers.

The use of Mask Generators and corresponding PN Generators importantly reduces the problem of computing masks to the problem of computing states in a LFSR. Chip-accurate masks are computed from the state of the LFSR. New masks are calculated only when a new offset is required, and computed masks can be stored for later use.

Figure 3 shows a second embodiment of an I PN generator for computing a mask in a shortest possible time albeit at the expense of the most hardware complexity. Less ROM storage is required and higher precision masks are produced. This hardware is

duplicated for a Q sequence PN generator Hardware 300 consists of ROM 301, control block 302, mask register 303, fifteen (15) mask circuits 304a-o, PN Generator A 305, and PN Generator mask circuit 306. Each of the mask circuits 304a-o is equivalent to mask circuit 100 of Figure 1. The PN generator requires 15 bit counters. ROM 301 has 3375 bits since it is required to store 225 words. New masks are generated by reading from 255 word ROM 301 with instructions stored in control block 302. One of the 15 stored sets is a sequence of masks resulting in shifts of $\{2^{14}, 2^{14}+1, 2^{14}+2, 2^{14}+3, 2^{14}+4, 2^{14}+5, 2^{14}+6, 2^{14}+7, 2^{14}+8, 2^{14}+9, 2^{14}+10, 2^{14}+11, 2^{14}+12, 2^{14}+13, 2^{14}+14\}$.

These stored elements are also represented as:

$$\textit{number of stored words and resulting shifts} = \sum_{i=0}^{14} \{\sum_{j=0}^{14} (2^i = j)\} .$$

Fifteen words are read from ROM 301 on each iteration for input into mask circuits 304a-o. Single bit outputs from each of mask circuits 304a-o are reloaded into mask register 303. During the next iteration, all fifteen mask circuits 304a-o produce a shift relative to a previous iteration. Thus, a chip accurate mask requires only fifteen sets within fifteen iterations. Control block 302 anticipates a binary representation of a required offset in order to simplify software interfacing. A state machine performs fifteen iterations of matrix multiplication and then enables output of computed masks for use in PN Generator mask circuit 306. This embodiment utilizes stored states of a linear feedback shift register for mask computing. The masking performance of this embodiment is shown in Table 3. Only fifteen clock cycles are required.

Table 3

	ROM bits	Hardware Complexity	Hardware slewing cycles (max)	Computation Iterations
I-PN Code	3375	451*	0	15
Q-PN Code	3375	451*	0	15

*Estimate is a combination of AND and XOR gates. Since all masking techniques require a mask register, its complexity is not included.

Figure 4 shows a third embodiment of a PN Generator A with simplified hardware for mask computing. This embodiment sacrifices speed for decreased hardware, and requires 255 clock cycles making it the slowest of the options but reduces the hardware by a

factor of fifteen. Hardware 400 consists of ROM 401, control 402, Mask Generator A 403, mask circuit 404, PN Generator A 405, and PN Generator mask circuit 406. Mask generator A 403 replaces the mask register 303 of Figure 3. Software instructions in control block 402 are more complex for this embodiment than for Figure 3. ROM 401 has 255 bits since it is required to store only 15 words. Mask Generator A 403 is masked with mask circuit 404 to produce a shifted sequence. Output from mask circuit 404 is another mask state after fifteen cycles, whereupon a different word is read from ROM 401. A new chip accurate mask is computed after 255 iterations, and the entire range of 2^{15} sequences are accommodated with 15 shifts each having a different offset power of 2. The stored elements are represented as:

$$\textit{number of stored words and resulting shifts} = \sum_{i=0}^{N} \{2^i\}$$

The masking performance of this embodiment is shown in Table 4.

Table 4

	ROM bits	Hardware Complexity	Hardware slewing cycles (max)	Computation Iterations
I-PN Code	225	36*	0	225
Q-PN Code	225	36*	0	225

*Estimate is a combination of AND and XOR gates. Since all masking techniques require a mask register, its complexity is not included.

Figure 5 shows a fourth embodiment of the present invention that moves the mask computation into a signal processor. Hardware 500 includes PN Generator B 501, Digital Signal Processor (DSP) 502 as Mask Generator B, PN Generating mask circuit 503, and optional PN Generating mask circuits 504. (DSP 502 is preferably a digital signal processor with instructions for LFSR and Count Ones added.) This embodiment provides greatest flexibility for variation among the parameters of time, storage, and software. Advantageously, this embodiment further permits slewing, and can combine slewing and computational iteration without affecting system performance. Moreover, more efficient algorithms can be added without changes to the hardware.

Calculation of a chip-accurate mask can be performed on DSP 502 with Mask Generator B. The masks are loaded into a register for use in the mask circuit. The output of the mask circuit is a shifted PN sequence relative to PN Generator B. The 15 bit state of the mask generator determines the mask input to the mask circuit. By using the binary decomposition of the required offset, any mask can be calculated with the mask generator. That is, for sequences of length $2^N - 1$, N = 15, in a 15 bit word. Since the mask generator is an LSFR, it too can be shifted. If 15 shifts are possible (i.e. 2^{14}, 2^{13}, ... 2^1, 2^0), any of 2^N possible mask generator states are reachable within 15 iterations. This resulting mask generator state is then used in the mask circuit to shift the PN sequence by any 2^N offset. Thus, with 15 x 15 bit words plus 15 iterations, no inaccuracies result. Only fifteen bits of shifted output sequence are required to determine the state of the mask generator, so that only fifteen words or less are stored in ROM. Each word represents a mask producing a phase shift of a power of two. Fifteen bits of shifted output are stored with a single mask, and a mask generator state is computed from these fifteen bits. Another word is read from ROM to generate a shifted output relative to a previously determined state. This masking technique is applicable to large shifts for computing the effective state of Mask Generator B. If DSP 502 runs Mask Generator B to compute small shifts, the corresponding masks are not required to be stored in ROM. The PN sequence of PN Generator B is never invalid due to mask inaccuracies, and therefore it is not necessary to slew the hardware of PN Generator B. Thus, this embodiment permits a trade-off between speed and storage size, a few examples of which are shown in Table 5:

Table 5

	ROM bits	Hardware Complexity	Hardware slewing cycles (max)	Computation Iterations
I-PN Code	225	30*	0	240
Q-PN Code	225	30*	0	240
I-PN Code	150	30*	16	176
Q-PN Code	150	30*	16	176
I-PN Code	135	30*	32	176
Q-PN Code	135	30*	32	176

*Estimate is a combination of AND and XOR gates. Since all masking techniques require a mask register, its complexity is not included.

The first two rows of Table 5 represent the limiting case of storing all possible phase shifts and not employing mask generator iteration for small shifts. Here, shifts of a few chips require their own masks stored in ROM. After generating a 15 bit output stream from masking Generator B, an extra cycle may be needed to reconstruct the state of Mask Generator B. The total number of computation iterations for the fifteen masks are (15 + 1)(15 masks), or 240 iterations.

The third and fourth rows represent a trade-off between storage size and mask generator iterations. Here, only 10 masks are stored in ROM, with resultant inaccuracies of up to 16 chips. These inaccuracies are compensated by iterating a Mask Generator B for up to 16 iterations. The total number of computation iterations are, therefore, [(15 + 1 iterations)(10 masks) + 16], or 176 iterations. The state of Mask Generator B is thus a chip accurate mask.

The fifth and sixth rows show another trade-off between storage size and software slews. Here, nine masks are stored, with resultant inaccuracies of up to 32 chips. These inaccuracies are compensated by iterating Mask Generator B for up to 32 iterations. The total number of computation iteration are, therefore, [(15 + 1 iterations)(9 masks) + 32 slews], 176 iterations. Again, this results in a chip accurate mask.

Figure 6 shows a fifth embodiment of the present invention introducing a new design for a PN generator to eliminate implicit 15 bit counters to handle masked sequence zero insertion. This embodiment uses only one mask circuit to compute states for the target PN generator. The entire PN generator is set to the PN offset and mask circuits are removed. The

hardware only performs zero insertion in the target PN generator, which eliminates the need for 15 bit counters to perform zero insertion in a masked output sequence.

The embodiment of Figure 6 includes Digital Signal Processor (DSP) 601, PN Generating mask circuit 602, reference PN Generator B 603, control block 604, target PN Generator 605, Exponent Counter 606, Walsh Generator 607, System Timer 608, and offset 609. DSP 601 computes chip accurate masks, and there is only one mask circuit 602 and one PN Generator 605. (DSP 601 is preferably a digital signal processor.) All masks are computed as shifts relative to reference PN Generator 603, which is utilized as a reverse link. Control block 604 shifts the output of mask circuit 602 to target PN Generator 605. Feedback is disabled while shifting is performed. Feedback is connected once shifting is performed, and LFSR of target PN Generator 605 activates.

Exponent Counter 606 performs two functions: (a) to accommodate the case where a mask shifts the PN state past the zero insertion point, and (b) for Walsh sequence generation. An overflow signal indicates a zero insertion point to control block 604. The corresponding integer value of the phase offset is added to the value of System Timer 608 upon loading a new phase offset. Walsh Generator 607 performs a mask of Exponent Counter 606 to generate a Walsh sequence by loading a representation of a row in the Walsh matrix. Software loads a binary representation of a row in the Walsh matrix.

The entire mask calculation is again performed on the DSP 601 acting as Mask Generator B. Various trade-offs between speed and storage space are available as in the fourth embodiment. The elimination of fifteen bit counters in each PN generator results in a decrease in speed. A chip-accurate mask is loaded into the mask circuit 602. Fifteen hardware cycles must elapse before a useable state is shifted out of the mask circuit 602. The number of shifting cycles is added to the number of computational iterations to obtain the total speed of computing a new mask. Shifting cycles performed in the hardware are wasted, and every reload of the PN Generator results in fifteen lost cycles because the output of the PN Generator is invalid for these cycles.

Masking performance, including several examples of the trade-off between speed and storage space, is shown in Table 6.

Table 6

	ROM bits	Hardware Complexity	Mask Gen.B Interations for Small Shifts	Computation Iterations
I-PN Code	165	30*	8	184
Q-PN Code	165	30*	8	184
I-PN Code	150	30*	16	176
Q-PN Code	150	30*	16	176
I-PN Code	120	30*	64	192
Q-PN Code	120	30*	64	192

*Estimate is a combination of AND and XOR gates. Since all masking techniques require a mask register, its complexity is not included.

The first and second rows represent a case of storing eleven possible phase shifts. The error of storing eleven of 14 shifts is 8 chips. The total number of computation iterations are [(15 + 1 iterations)(11 masks) + 8 iterations], or 184 iterations. The total time required to load a new state is [(15 + 1 iterations)(11 masks) + 8 iterations + 15 shifts], or 199 cycles.

The third and fourth rows represent a case of storing 10 masks in ROM. The error of storing 10 of 14 shifts is 16 chips. The total computation iterations are [(15 + 1 iterations)(10 masks) + 16 iterations], or 176 iterations. The total time to compute a new mask is [(15 + 1 iterations)(10 masks) + 16 iterations + 15 shifts], or 191 cycles.

The fifth and sixth rows represent a case of storing 8 masks in ROM. Th error of storing only 8 of 14 shifts is 64 chips. The total computation iterations are [(15 + 1 iterations)(8 masks) + 64 iterations], or 192 iterations. The total time to shift the PN Generator is [(15 + 1 iterations)(8 masks) + 64 iterations + 15 shifts], or 207 cycles.

This sixth embodiment permits more intelligence in computation of masks. Once a more efficient method of mask computation is determined, e.g. reconfiguring a feedback polynomial, the system is enhanceable by changing program code within DSP 601.

The preferred embodiment described in the preceding description is provided to enable one of ordinary skill in the art to make and use the invention. Various modifications to these embodiments will be readily apparent to those having ordinary skill in the art. Therefore, the present invention is not intended to be limited to the specific embodiments

described herein, but is to be accorded the widest scope consistent with the principles and novel features disclosed herein.

CLAIMS:

1. A system for generating mask polynomials in code division multiple access communications, comprising:

a pseudo-random noise generator for generating first polynomials, said first polynomials representing a pseudo-random noise code sequence of a communication channel;

a mask generator for generating second polynomials, said second polynomials being derived from said first polynomials, said second polynomials permitting calculation of masks representing any desired phase shift offset in said sequence.

2. The system for generating mask polynomials of claim 1, wherein:

said mask generator has a plurality of mask circuits.

3. The system for generating mask polynomials of claim 1, further comprising:

a Walsh generator and an exponent counter.

4. The system for generating mask polynomials of claim 1, wherein:

said first and second polynomials have complementary Galois and Fibonacci configurations.

5. The system for generating mask polynomials of claim 1, further comprising:

means for integrating over at least a portion of said sequence to determine whether said pseudo-random code sequence is present, wherein a result of an integration is zero when said pseudo-random code sequence is not present and non-zero when said pseudo-random code sequence is present.

6. The system for generating mask polynomials of claim 1, wherein:

said pseudo-random noise generator is a linear feedback shift register or a linear sequence shift register.

7. The system for generating mask polynomials of claim 1, wherein:

said pseudo-random noise generator is slewed by varying the clock rate.

8. The system for generating mask polynomials of claim 1, wherein:

a ROM for storing words to produce said masks of said phase offset in said sequence, said ROM storing N words to produce 2^N of said phase shift offsets.

9. A method for generating mask polynomials, comprising:

generating a pseudo-random noise sequence with pseudo-random noise generator;

storing N words in ROM;

generating 2^N masks with a mask generator from said N words;

shifting a phase offset of said pseudo-random sequence with said 2^N masks;

integrating at least a portion of a pseudo-random noise sequence.

10. The method for generating mask polynomials of claim 9, further comprising:

calculating said phase offset from 0 to 2^N - 1 within N iterations.

11. The method for generating mask polynomials of claim 9, further comprising:

slewing said phase offset by varying the clocking rate.

12. The method for generating mask polynomials of claim 9, further comprising:

calculating a mask for small shifts by iterations of said mask generator.

13. An apparatus for generating mask polynomials in code division multiple access communications, comprising:

pseudo-random noise generator means for generating first polynomials representing a pseudo-random noise code sequence;

mask generator means for generating second polynomials permitting calculation of masks representing any desired phase shift offset.

14. An apparatus for generating mask polynomials, comprising:

means for generating a pseudo-random noise sequence;

means for storing N words:

means for generating 2^N masks from said N words;

phase offset shifting means for shifting a phase offset of said pseudo-random sequence;

integrating means for integrating at least a portion of said pseudo-random sequence.

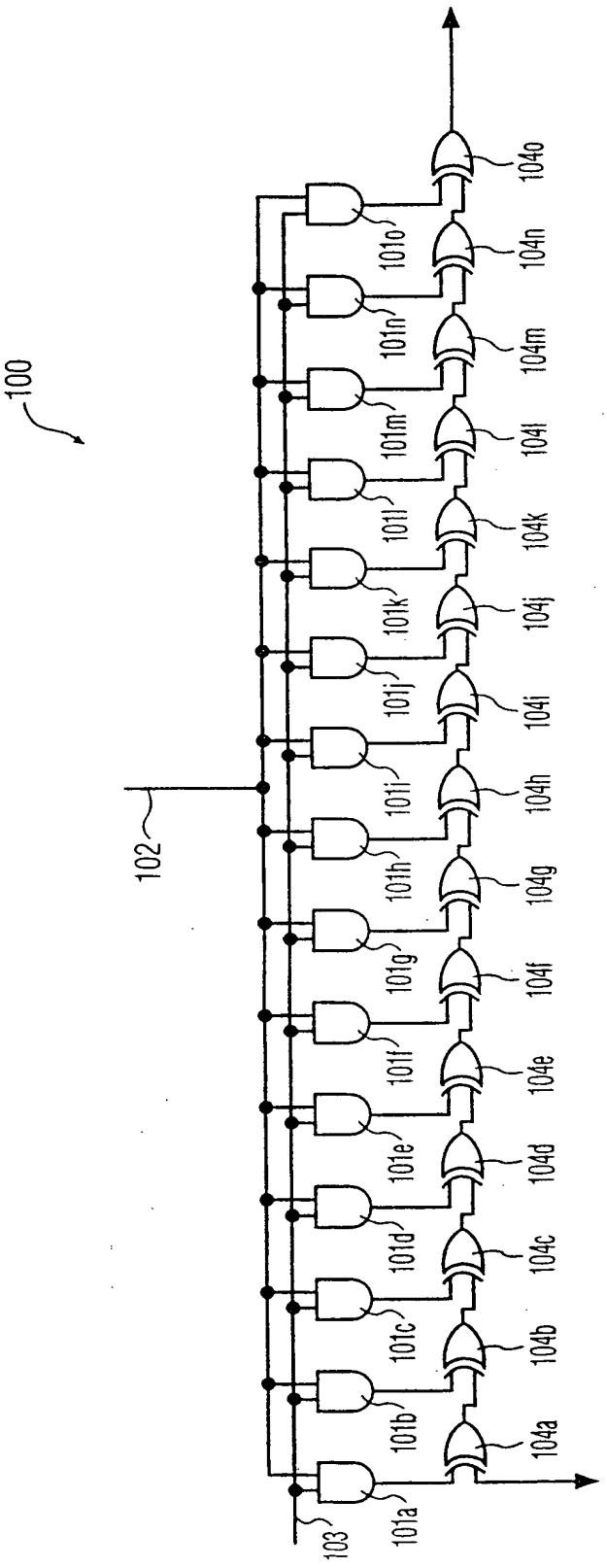

100
101a
101b
101c
101d
101e
101f
101g
101h
101i
101j
101k
101l
101m
101n
101o
102
103
104a
104b
104c
104d
104e
104f
104g
104h
104i
104j
104k
104l
104m
104n
104o

FIG. 1

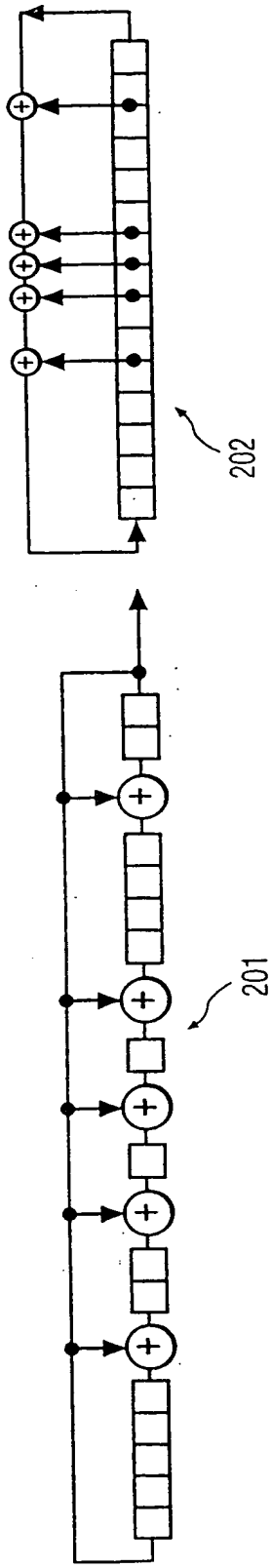


FIG. 2A

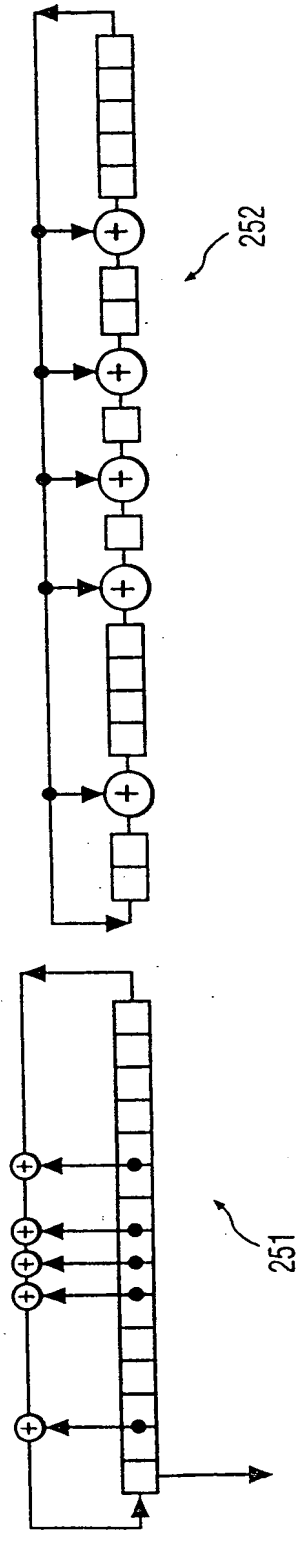


FIG. 2B

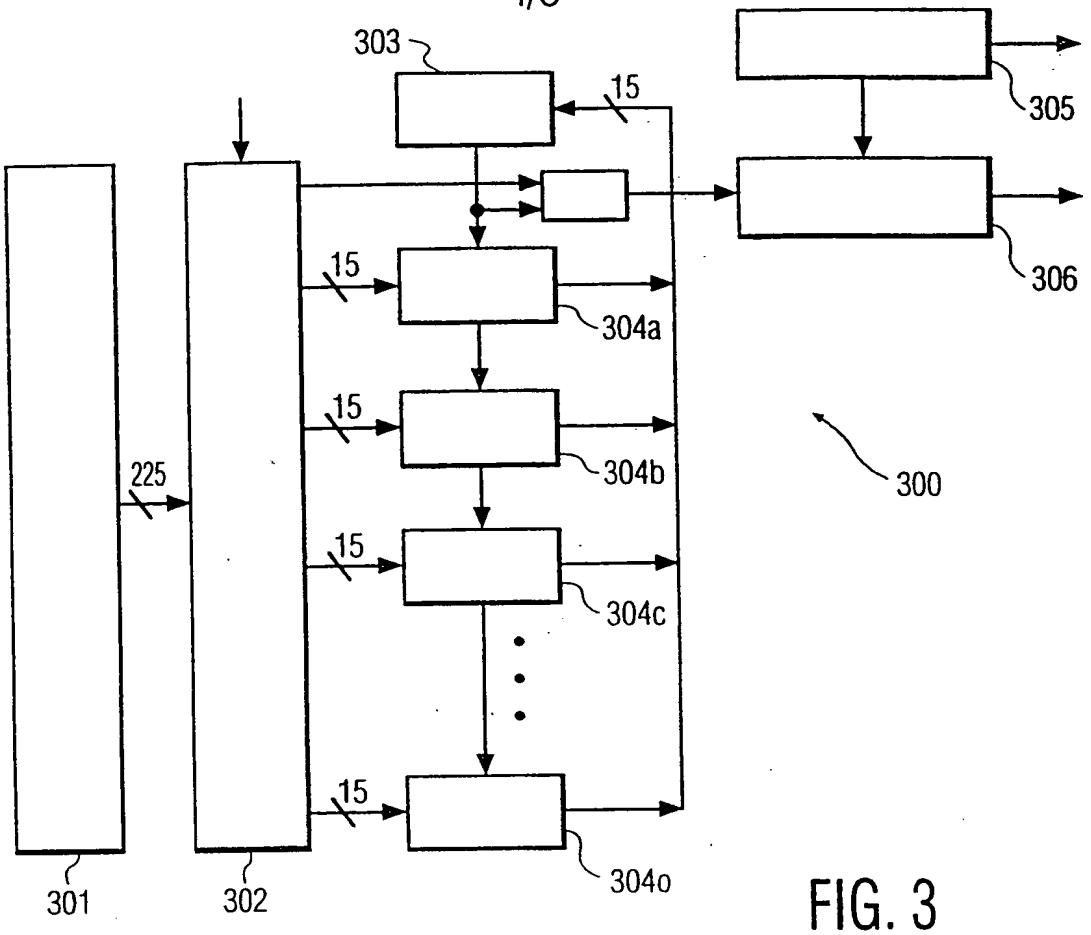

303
15
305
306
15
304a
15
304b
225
15
304c
15
304o
300
301
302

FIG. 3

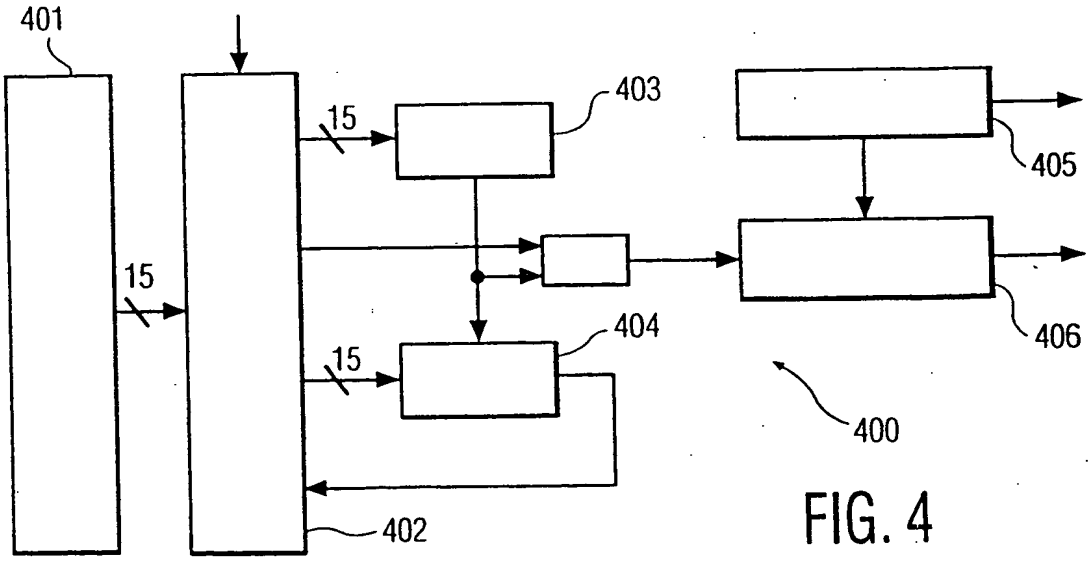

401
403
15
405
15
404
406
15
400
402

FIG. 4

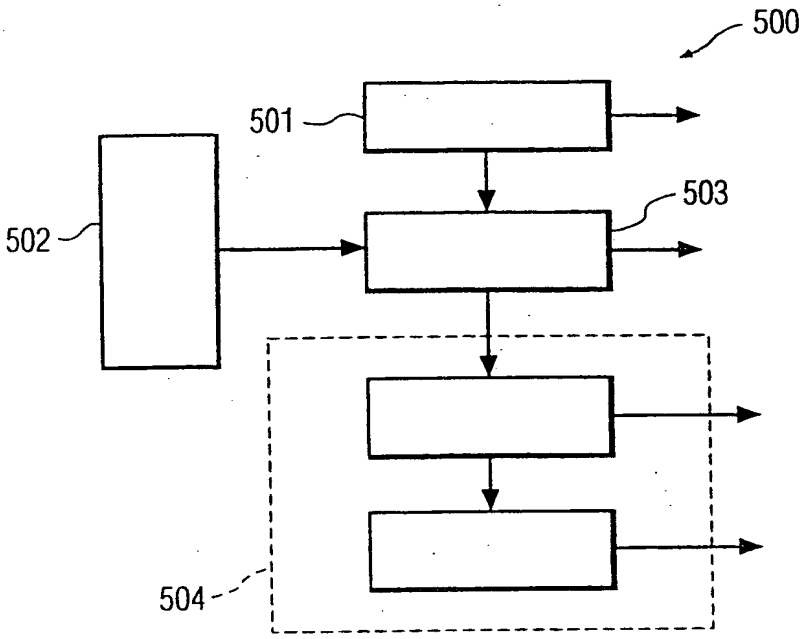


FIG. 5

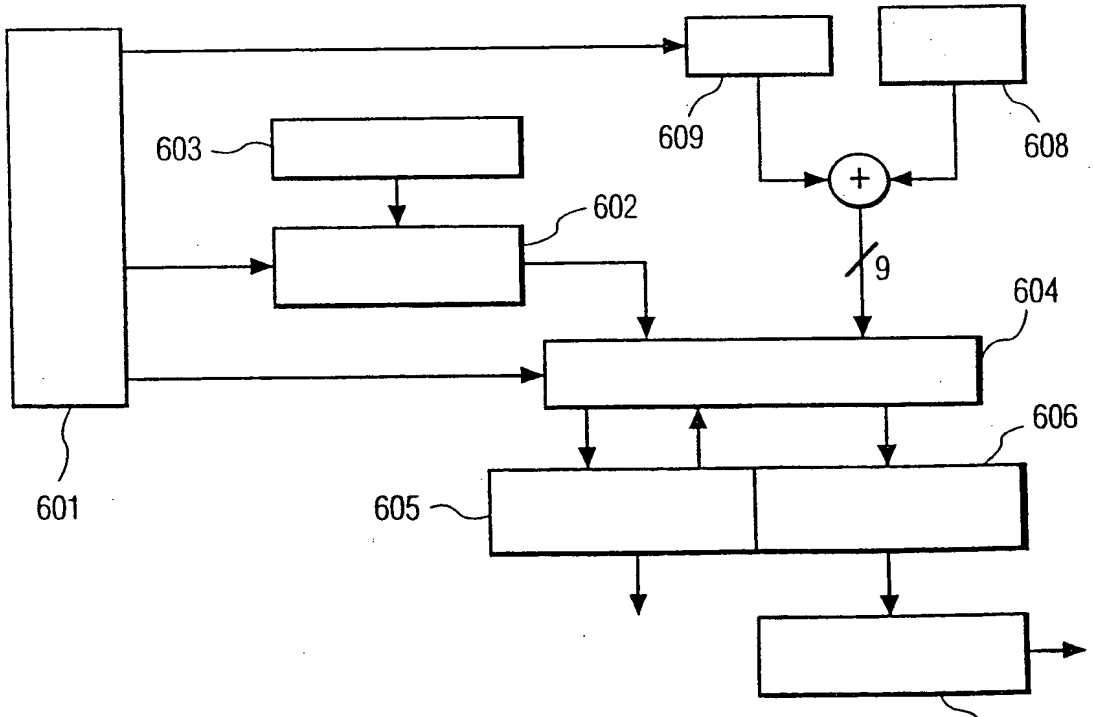


FIG. 6